BCNF Decomposition

Example: Decompose into BCNF - **Restaurant(id, name, rating, popularity, rec)**

1. **id → name, rating**
2. **rating → popularity**
3. **popularity → rec**

Given R(A, B, C, D, E), and functional dependencies: A → C, BD → A, D → E

1. Find the following closures: {A}+, {B}+, {D}+, and {BD}+

2. Decompose R into BCNF. In each step, explain which functional dependency you used to decompose and explain why further decomposition is needed. Your answer should consist of a list of table names and attributes. Make sure you indicate the keys for each relation.

Relational Algebra

RA Operators:

σ =

π =

⋈ =

ɣ =

δ =

Example: Make this SQL query into RA (remember FJWGHOS)

```
SELECT R.b, T.c, max(T.a) AS T_max
  FROM Table_R AS R, Table_T AS T
 WHERE R.b = T.b
 GROUP BY R.b, T.c
HAVING max(T.a) > 99
```

Convert the following SQL queries into logical RA plans, given the following schemas:
Actor(aid, fname, lname, age)
ActsIn(aid, mid)
Movie(mid, name, budget, gross)

1.
```
SELECT A.fname, A.lname, A.age
FROM Actor AS A
WHERE A.fname = 'Patrick'
      AND A.lname = 'Stewart';
```

2.
```
SELECT M.name, COUNT(*) AS cnt
FROM Actor AS A, ActsIn AS AI, Movie AS M
WHERE A.aid = AI.aid AND M.mid = AI.mid
      AND A.age < 30
GROUP BY M.mid, M.name
HAVING COUNT(*) > 1;
```

3.
```
SELECT A.aid
FROM Actor AS A
WHERE A.aid NOT IN (
      SELECT AI.aid
      FROM ActsIn AS AI, Movie AS M
      WHERE AI.mid = M.mid AND AI.aid = A.id AND M.name = ‘Star Wars’
);
```

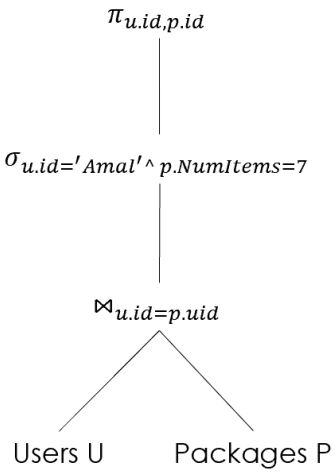
RA to RA

Consider the fact that Amazon has shipped several billion packages over the course of its >20y history and that it may surpass 10B packages by 2030. Assume that it tracks its packages and users using the following schema:

Packages(<u>PackageID</u>, UserID, DestAddress, NumItems)

Users(<u>UserID</u>, CreditCardNumber, Languages)

Now, consider the following RA tree:



You may notice how, although the PACKAGES table is very very large (10B!!), an individual user may have a very small number of rows. Generate a logically-equivalent tree which, ideally, takes advantage of this fact.